Requested Patent: JP7200317A

Title: OPERATION RIGHT MANAGEMENT EQUIPMENT ;

Abstracted Patent: JP7200317 ;

Publication Date: 1995-08-04 ;

Inventor(s): TAKAHASHI TOSHINARI; others: 04 ;

Applicant(s): TOSHIBA CORP ;

Application Number: JP19930349335 19931228 ;

Priority Number(s): ;

IPC Classification: G06F9/46; G06F1/00; G06F12/14 ;

Equivalents: ;

ABSTRACT:

PURPOSE:To attain flexible operation right management for threads by providing a means storing revision right information and a means referencing the stored revision right information before execution of revision of operation right information and verifying whether or not the revision is permitted to the equipment.

CONSTITUTION:The equipment is provided with a means storing revision right information representing whether or not revision of operation right information is permitted based on a thread or the user being a subject of the revision or a memory area or a program in which the subject of revision is in existence and a means referencing the stored revision right information before execution of the revision of the operation right information and verifying whether or not the revision is permitted. That is, an operation right list storage section 7 stores a table relating to thread protection and storing a fact of the right executing the operation to each thread and used for an operation right discrimination section 12. Thus, not only the management of the operation right to the thread but also the revision right is flexibly expressed and managed by the combination of various conditions.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平7－200317

(43)公開日　平成７年(1995)８月４日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	9/46	３４０　Ｂ	7629－5Ｂ		
	1/00	３７０　Ｅ			
	12/14	３１０　Ｋ			

審査請求　未請求　請求項の数３　ＦＤ　（全 14 頁）

(21)出願番号　特願平5－349335

(22)出願日　平成５年(1993)12月28日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　高橋　俊成
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

(72)発明者　岡本　利夫
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

(72)発明者　福本　淳
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

(74)代理人　弁理士　鈴江　武彦

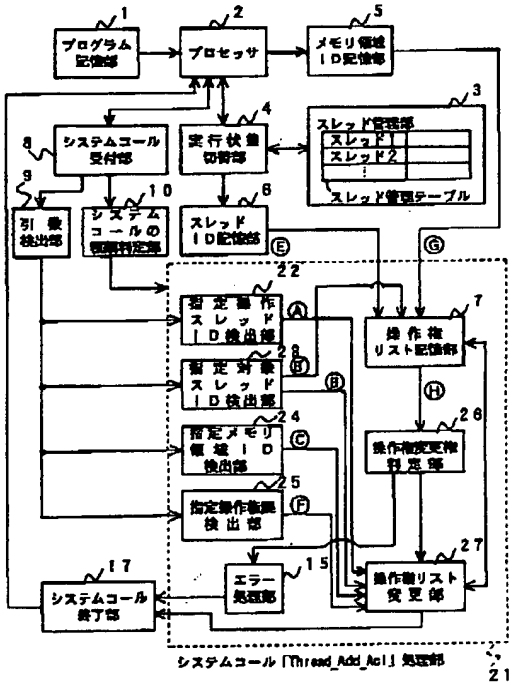
最終頁に続く

(54)【発明の名称】　操作権管理装置

(57)【要約】

【目的】　スレッドの操作権管理の柔軟化を目的とする。

【構成】　本発明は、複数の並行して実行されるスレッドによりプログラムを実行する手段と、各スレッドに対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを示す変更権情報を記憶する手段と、前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする。



1

【特許請求の範囲】

【請求項1】複数の並行して実行されるスレッドによりプログラムを実行する手段と、

各スレッドに対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、

前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを示す変更権情報を記憶する手段と、

前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする操作権管理装置。

【請求項2】実メモリまたは仮想メモリを複数のメモリ領域の集合として管理する手段と、

各メモリ領域に対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、

前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを示す変更権情報を記憶する手段と、

前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする操作権管理装置。

【請求項3】前記変更権情報を前記操作権情報に付随されて記憶することを特徴とする請求項1または2に記載の操作権管理装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、計算機システム中でのメモリ領域またはスレッドへの操作権を管理する操作権管理装置に関するものである。

【0002】

【従来の技術】近年マイクロプロセッサの性能向上により複数の小型のコンピュータをネットワークでつないで使用することが通常のこととなり、ネットワーク結合された計算機やユーザー同士での協調作業が可能となり、操作権の管理を厳密に行う必要が出てきた。

【0003】またオブジェクト指向技術の進歩やサーバ・クライアント・モデルのような新しいパラダイムの出現により、従来の記憶装置やファイルを単位とした操作

2

権管理では十分な性能が得られなくなってきた。

【0004】こういった問題を解決するために、単一仮想記憶などの技術により、メモリやCD－ROMやハードディスク・ドライブのような記憶装置、さらにはディスプレイやキーボードといったあらゆる計算機資源を同一の概念で管理するオペレーティング・システムの設計が試みられている。中でも、各種の資源を同一の仮想空間上に配置する単一仮想記憶の技術は特徴的なものである。さらに、この単一仮想記憶をいくつかのブロックに分割し、ブロック単位で操作権を管理する方法が提案されている。この管理単位を「メモリ領域」と呼び、あらゆる記憶装置の操作権管理の単位として用いられる。

【0005】一方、オペレーティング・システムはマイクロ・カーネル技術の定着により、プログラムの実行を管理するスレッドという単位を使って管理するのが一般的になりつつある。すなわち、プログラム実行の制御対象としてプロセスまたはスレッドと呼ばれる制御単位が用いられる。個々のスレッドは独立にプロセッサのレジスタ値やスタックなどを持ち、オペーレーティング・システムによりプロセッサを割り当てられた間だけプログラムを実行し、一定時間後またはハードウェア割り込みなどが発生したときにオペレーティング・システムはスレッドに対するプロセッサの割り当てを解除して他のスレッドへプロセッサを割り当てる。個々のスレッドは独立したレジスタ値を持っているため、仮想的に並列に動いているプロセッサであるとみなすことができる。

【0006】このようなオペレーティングシステムでは、メモリ領域とスレッドという二つの概念を単位として計算機資源を構成し、そこへの操作権を管理する情報もACL (Access Control List)という形で各メモリ領域および各スレッドに付随させて管理することが行われる。

【0007】例えば、スレッドの場合を例に説明すると、スレッド自体を操作したいとき、すなわちスレッドの実行を一時停止したり再開したりしたいときやスレッドを強制的に消滅させたいとき、またはスレッドの情報（レジスタ値やスタック等の内部状態）を調べたいときのため、オペレーティングシステムはスレッド操作命令群を備えており、あるスレッドから別のスレッドを操作することが可能である。このようなスレッド操作命令群は、ユーザプログラムからシステムコールとしてオペレーティング・システムに指示し、利用できるようになっている。

【0008】このようなスレッド操作命令があるときに、どのスレッドからでもスレッド操作命令を発行できるようにすると、不当な操作もできることになってしまうので保護の機能が必要になる。たとえばスレッドを強制的に消滅させるようなことは特定の権限を持つスレッドだけから行えるようにすることが必要になる。

【0009】さらに、仮想空間が複数あり、仮想空間ご

3

とに1つのプログラムやデータが配置されており、スレッドもその空間内しか動きまわれないようなＭａｃｈ（参考文献：「分散オペレーティング・システム」前川守他編、共立出版）をはじめとするオペレーティング・システムの場合、同じ空間内に存在するスレッドの保護を同一とし、スレッドの保護をメモリ空間の保護に依存する方法をとっている。

【００１０】しかし、それらのスレッドを区別して保護することはできない。また、単一仮想空間のシステムのように、１つのアドレス空間内にすべてのプログラムが配置され、複数のスレッドが存在する場合、スレッドの保護はメモリ保護だけではなくオペレーティングシステムによる権限の設定が必要になる。

【００１１】この問題のひとつの解決法は、スレッドの所有者を決めて、同一の所有者のスレッド間ではスレッドの操作を許す方式であり、Unixオペレーティングシステムのsignalはこれに相当する。しかしこれでは同一の所有者のスレッド間の保護の柔軟性に欠け、さらに、操作権限を他の所有者のスレッドに渡すことができない。また、複数の所有者のスレッドからのスレッド操作を可能にしたりできないなど、柔軟にスレッドの操作ができるようなスレッド保護機能が不足していた。

【００１２】また、一方ではデータ・ファイルへの操作権の多彩な設定ができるように、ファイルやディレクトリに対するＡＣＬに、情報の追加権、削除権、ＡＣＬの変更権などのさまざまな情報を付加して柔軟性を高めようとするＤＣＥ（参考文献：ＯＳＦ　ＤＣＥ技術解説、ソフトリサーチセンター）のような試みもあるが、やはり複数の所有者にメモリ領域のＡＣＬの管理を行わせるような柔軟な保護機能は不足している。

【００１３】

【発明が解決しようとする課題】以上述べてきたことに基づいて従来の操作権管理装置の主要な課題をまとめると次のようになる。

【００１４】（１）従来、スレッドに対する操作を、この操作を行う主体や、この操作を行う主体が存在するメモリ領域などに基づいて、許可するか否かを指示する方法は試みられている。しかし、スレッドに対するこの操作権情報を変更する権利については、例えば、そのスレッドの所有者には認める、あるいはルートと呼ばれる特別のユーザにのみ認める、といった単純な管理しか行うことができなかった。

【００１５】（２）また、従来、実メモリ、仮想メモリ、データ・ファイルなどのメモリ領域に対する操作を行う主体や、この操作を行う主体が存在するメモリ領域などに基づいて、許可するか否かを指示する方法は試みられている。しかし、メモリ領域に対するこの操作権情報を変更する権利については、例えば、そのメモリ領域の所有者には認める、あるいはルートと呼ばれる特別のユーザにのみ認める、といった単純な管理しか行うことができなかった。

4

【００１６】（３）さらに、スレッドやメモリに対する操作権を柔軟に記述し、かつ、同様に変更権をも柔軟に記述すると、柔軟な記述ができるようになる反面、スレッドやメモリに対する保護のためのデータ量が多くなり、その管理が複雑になるという欠点を持つという問題がある。

【００１７】このように、従来の操作権管理装置では、メモリ領域単位での操作権限の設定ができなかったり、操作権限の設定の面で柔軟性に欠けたり、または、同一の仮想アドレス空間を共有しているスレッド間では保護が無かったか、または制約が強かった。

【００１８】本発明は、上記課題を考慮してなされたものであり、別々の仮想空間にいるスレッド間はもとより、同一仮想アドレス空間を共有しているスレッド間でのスレッド操作命令あるいは個々のメモリ領域に対して、同一の方法で柔軟性があり、かつ十分な保護を行う操作権管理装置を提供することを目的とする。

【００１９】

【課題を解決するための手段】上記目的を達成するために、本発明（請求項１）では、複数の並行して実行されるスレッドによりプログラムを実行する手段と、各スレッドに対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを示す変更権情報を記憶する手段と、前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする。

【００２０】また、本発明（請求項２）では、実メモリまたは仮想メモリを複数のメモリ領域の集合として管理する手段と、各メモリ領域に対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを示す変更権情報を記憶する手段と、前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする。

【００２１】また、望ましくは、上記発明（請求項１）

5

において、スレッド作成時には、あらかじめ定められたデフォルトの変更権情報が設定されることを特徴とする。

【００２２】また、望ましくは、上記発明（請求項２）において、メモリ領域作成時には、あらかじめ定められたデフォルトの変更権情報が設定されることを特徴とする。

【００２３】さらに、上記各操作権管理装置において、前記操作権情報を記憶する手段の情報を変更することが不可能になるような、前記変更権情報を記憶する手段の情報の変更を認めないようにしても良い。

【００２４】また、本発明（請求項３）では、上記発明（請求項１または２）において、前記変更権情報を前記操作権情報に付随されて記憶することを特徴とする。

【００２５】さらに、上記各操作権管理装置において、前記操作権情報を記憶する手段の情報を変更することのできる複数の条件を記憶する操作権管理権限記憶手段と、前記変更権情報を記憶する手段の情報に変更を加えようとする際に該操作権管理権限記憶手段の情報によりこの変更を行う権限があるか否かを検証する手段と、この権限があると検証された場合に前記変更を許可する手段とをさらに設けても良い。また、この場合、前記変更権情報を記憶する手段の情報を変更することが不可能になるような、前記操作権変更権管理権限記憶手段の情報の変更を認めないようにしても良い。

【００２６】

【作用】本発明（請求項１）によれば、スレッドへの操作権の管理のみならず、その操作権の管理の権限である変更権についても、さまざまな条件の組み合わせによって柔軟に表現、管理することができるようになる。

【００２７】この機構を用いれば、スレッドの操作権に関して、特定のスレッドにのみその変更を許可したり、またそれらの複数の条件を組み合わせて表現したりすることができる。

【００２８】特に、近年多く利用されるようになったグループウェアと呼ばれる計算機の協調作業アプリケーションを作成する際には、スレッドの共有や、排他制御、他のプログラムからの保護、などのきめ細かな制御を容易に記述することが必要になるが、本発明を用いればこの記述を柔軟に行うことができる。

【００２９】また、プログラムの誤りや、システム・コールの不正な呼び出しなどによって、スレッドへの操作権が異常に書き換えられてしまうことを未然に防ぐことができるので、スレッドの共有を行いつつもなおスレッドの動作に関する信頼性が飛躍的に向上する。

【００３０】また、本発明（請求項２）によれば、メモリ領域への操作権の管理のみならず、その操作権の管理の権限である変更権についても、さまざまな条件の組み合わせによって柔軟に表現、管理することができるようになる。

6

【００３１】この機構を用いれば、メモリ領域への操作権に関して、特定のスレッドにのみその変更を強化したり、特定のメモリ領域に存在するスレッドにのみその変更を許可したり、またそれら複数の条件を組み合わせて表現したりすることができる。

【００３２】特に、近年利用が試みられている単一仮想記憶と呼ばれる、複数のアプリケーションが同一の仮想空間上で動作するオペレーティング・システムにおいては、メモリ領域の共有や、排他制御、他のプログラムからの保護、などのきめ細かな制御を記述することが必要となるが、本発明を用いればこの記述を柔軟に行うことができる。

【００３３】また、プログラムの誤りや、システム・コールの不正な呼び出しなどによって、メモリ領域への操作権が異常に書き換えられてしまうことを未然に防ぐことができるので、メモリ領域に書かれたデータが不用意に書き換えられてしまうことがおこらず、メモリ領域の共有を行いつつもなおメモリ領域に書かれたデータに対する信頼性が飛躍的に向上する。

【００３４】さらに、本発明（請求項３）によれば、スレッドやメモリに対する操作権情報と、その操作権情報の変更を行う変更権情報とは、一般に共通点が多く、データ構造も似通ったものになることが多い。本発明においては、変更権情報を操作権情報に付随させて記憶することにより、共通のデータをなるべく一つのものとして管理するため、データ量を少なくすることができ、変更権および操作権を検証するための手続きも共通化する。

【００３５】また、スレッドやメモリに対する操作権や変更権などの管理は、一般にメモリ管理ユニットと呼ばれるハードウェアを用いることによって、実行性能を向上させることができる。そのような場合に、操作権および変更権それぞれに別のハードウェアを用意することは、計算機の構成を複雑にする可能性がある。しかし、本発明によって、操作権と変更権を同一に管理することにより、必要なハードウェアを単純化することができる。

【００３６】

【実施例】以下、図面を参照して本発明の実施例について説明する。

【００３７】本発明に係る操作権管理装置は、オペレーティング・システム（ＯＳ）と呼ばれる計算機システム全体の実行制御を行うものの構成要素の一部である。ＯＳは計算機資源であるメモリ、プロセッサ、周辺機器を管理し、また、ユーザプログラムとよばれるユーザが指示した実務上の処理を行うプログラムの実行を制御する。一つの装置上には複数のユーザプログラムがのっており、ＯＳがそれらのプログラム間の制御も行う。

【００３８】ユーザプログラムは、実行上に必要となる物理メモリなどの資源を確保したり、周辺機器にデータを入出力したり、他のユーザプログラムにデータを転送

7

したり、処理を依頼したり、実行を制御するためにＯＳに対して指示を行うことが必要となる。このＯＳに対する指示をシステムコールを呼ばれる特別な手段で行う。本実施例では、システムコールのうち、操作権を考慮するユーザプログラムの制御に関して説明する。

【００３９】操作権については、スレッドの管理とメモリ領域の管理がある。これらメモリ領域の管理とスレッドの管理については同一の部分が多いので、同様の機構を重複して説明することは避け、主にスレッドに対する操作権管理を例として、同一の部分には同一の用語を用いて説明する。

【００４０】なお、本実施例では、メモリの「操作」とは、アクセス（読み書き実行）のことを表すものとする。

【００４１】最初に、本実施例の特徴を概略的に述べる。本実施例では、個々のメモリ領域またはスレッドに対して、どのような条件で操作を許可するかという情報を登録したリストを設け、このリストにより許可されている場合に限り、操作を許可するとともに、この情報自体を変更することのできる権限を、前記リストと同一の形式、類似の形式あるいは共通の形式で管理する。また、特定の条件の時にのみこの変更を許す機構を付加する。また、この変更をする権限の変更についても同一の形式、類似の形式あるいは共通の形式で管理する。

【００４２】この結果、本実施例は、不当な操作命令やプログラムの誤りからメモリ領域やスレッドを保護し、またこのような誤った操作命令を検出することによってプログラムの誤りの検出を容易とするために、メモリ領域またはスレッドごとに設けた操作の可否を示す情報を変更する権利自体もそのメモリ領域またはスレッドが持つので、操作権の管理のみならず、操作権の管理の権限すらも同一の形式、類似の形式あるいは共通の形式で管理し、削除し、追加し、または変更することを柔軟に行うことができるようになる、という作用効果を奏するものである。

【００４３】また、この操作権の管理の権限は、メモリ領域やスレッドの所有者とは全く別の機構より、通常のアクセス制御リストと同様の形式で複数持つので、操作権管理の権限を複数のスレッドで共有したり、自分の作ったメモリ領域やスレッドの管理権を他のスレッドに譲渡したりするなどの柔軟な操作権管理が可能となる。

【００４４】（第１の実施例）以下、本発明の第１の実施例について説明する。

【００４５】図１は、本実施例の全体構成図である。プログラム記憶部１には、計算機で処理する命令が記述されたユーザプログラムや処理する対象のデータ、計算機で処理途中の結果を一時的に記憶するスタックなどのデータがメモリ空間上に蓄えられている。メモリ空間は、複数の領域（メモリ領域）に区切られて管理され、それぞれの領域には、お互いを区別するための識別子がついており、メモリ領域ごとにプログラムやデータやスタックを配置している。

8

【００４６】さらに、プログラム記憶部１に蓄えられているプログラムを解釈実行し、データを処理する実行主体であるプロセッサ２と、プロセッサ内の状態を管理するスレッド管理部３、プロセッサの実行状態の切り替え処理を行う実行状態切り替え部４がある。

【００４７】プロセッサ２内には、命令の実行途中に一時的に利用するレジスタ類、現在実行している命令のプログラム記憶部での位置を示すためのプログラムカウンタ（ＰＣ）、スタックデータの先頭位置を示すスタックポインタ（ＳＰ）などが存在し、これら一連のデータを入れ替えることで、複数の別々のユーザプログラムの処理を時分割で並行して進めることが可能となる。

【００４８】スレッドとは、上記プロセッサ内で使用する一連のデータのことを呼び、これを複数組用意し、このスレッドを切り替えてプロセッサにセットし実際に実行させることにより、時分割処理が可能となる。

【００４９】メモリ領域ＩＤ記憶部５は、現在実行中のプログラムがプログラム記憶部１のどのメモリ領域に存在しているかを、そのメモリ領域のＩＤとして記憶しているものである。これは、プロセッサ２内のＰＣの値を利用し、現在実行中のメモリ領域ＩＤを求めて、記憶している。このメモリ領域ＩＤは、現在実行中のプログラムＩＤとみなすこともできる。

【００５０】スレッド管理部３の中にはスレッド管理テーブルと呼ぶ表があり、プロセッサ２にセットするレジスタの情報やそのほかの情報がスレッドごとに管理されている。スレッドには、お互いを区別するために識別子（スレッドＩＤ）がついている。

【００５１】実行状態切替部４は、現在プロセッサ２上で実行しているスレッド（例えばスレッドＩＤ＝１）を中断し、スレッド（スレッドＩＤ＝１）の状態をスレッド管理テーブルの所定の位置（スレッドＩＤ＝１の格納領域）に退避し、別のスレッド（例えばスレッドＩＤ＝２）を起動すべく、スレッド管理テーブルのスレッド（スレッドＩＤ＝２）の情報をプロセッサ２に再セットし、スレッド（スレッドＩＤ＝２）の実行を再開させる。また、実行状態切替部４には、さらに、どのスレッドをセットし退避するかを決定するスレッドスケジューリング機構（図示せず）と、いつスケジューリングするか指示するタイマ部（図示せず）が存在する。

【００５２】スレッドＩＤ記憶部６は、スレッドＩＤを記憶するためのものである。

【００５３】操作権リスト記憶部７は、スレッド保護に関するテーブルを保持し、スレッドごとにスレッドに対する操作を実行する権限がどうなっているかを記憶するもので、操作権限判定部１２で使用される。本実施例はスレッドの操作権管理を例として説明しているが、メモリ領域の操作権管理の場合には、この操作権リスト記憶

部７はメモリ領域の保護に関するテーブルを保持し、スレッドごとにメモリ領域に対する操作を実行する権限がどうなっているかを記憶するものである。以下同様にスレッドに関する機構とメモリ領域に関する機構の同一の部分については説明を省略し、スレッドに関するものを代表して記述する。

【００５４】さて、システムコールとしてユーザプログラムを実行しているあるスレッドが他のスレッドに対して制御しようとする例として、Thread_Get _Statusシステムコールを発行する場合について説明する。

【００５５】このシステムコールは、スレッドＩＤで指示されたスレッドの状態、つまりそのスレッドのレジスタ類の情報を、このシステムコールの呼び出し元のユーザ・プログラムに通知するものである。いま、スレッド１（スレッドＩＤ＝１）がプログラムＡ（メモリ領域ＩＤ＝Ａ）を実行中、そこでスレッド２（スレッドＩＤ＝２）の情報をこのシステムコールで得ようとしたとする。Ｃ言語で記載されているプログラムＡでは、次のように記されている。

err = Thread_Get _Status(2, &Thread _Status);ここで、関数Thread_Get _Statusがスレッドの情報を得るシステムコールであって、第１引数の２は、スレッド２（スレッドＩＤ＝２）の情報を得ることを指示しており、第２引数の&Thread _Statusはシステムコールの結果得られたスレッド２の情報を格納するメモリ領域の先頭位置を指示している。また、このシステムコールの実行が成功したかどうかは、この関数の返り値として変数 err に入る。

【００５６】この関数をプログラムＡのスレッド１が実行すると、スレッド１はシステムコールにより実行モードがユーザレベルから特権レベルに遷移し、ＯＳを呼び出す。特権レベルとは、ＯＳが処理を行う専用レベルのことである。

【００５７】呼び出されたＯＳでは、ＯＳ内のシステムコール受付部８に処理が移る。ここでは、このシステムコールを呼びだした元のユーザプログラムの中断処理を行い、要求されたシステムコールの種類と指示された引き数を受取り、それぞれその情報を引数検出部９とシステムコールの種類判別部１０に送る。

【００５８】本例では、指示されたシステムコールは、Thread_Get _Statusなので、システムコールの種類判定部１０で、次には、Thread_Get _Status処理部１１を呼び出す。

【００５９】そこでの処理は、図２に示した操作権リスト記憶部７を使って行われる。この例では操作権リスト記憶部７はスレッドに対する操作権リストを記憶するものである。この記憶部には、図２に示したようなテーブルを保持している。テーブルには、操作スレッドＩＤ、対象スレッドＩＤ，操作スレッドが存在するメモリ領域ＩＤ、スレッド操作権限、スレッド操作権変更権限の各エリアをもっている。

【００６０】同様のエリアとしては、対象スレッドが存在するメモリ領域ＩＤを指定することも考えられるが、本例ではかかるエリアを持たないものを例として説明する。

【００６１】メモリ領域に対する操作権リスト記憶部も同様であるが、テーブルの内容は若干異なる。これについての詳細は第２の実施例で説明する。

【００６２】操作スレッドＩＤとはスレッド操作のシステムコールを実行しようとしているスレッドのスレッドＩＤのことで、対象スレッドＩＤとはこのシステムコールの引数で指定された操作対象のスレッドＩＤを示す。また、操作スレッドが存在するメモリ領域ＩＤとは、この実行されようとしているシステムコールを含むプログラムの存在するメモリ領域ＩＤのことであり、スレッド操作権限とは、どういうスレッド操作が可能かを示すもので、スレッドの内部状態を読み出す権限、スレッドの内部状態を変更する権限、スレッドの実行を停止／再開する権限の３つの権限を独立に指定することができる。スレッド操作権変更権限は、スレッド操作権限の記入されたフィールドの情報を変更する権限のことであり、この権限が指定されている場合、その列に書かれているスレッド操作権限のフィールドの情報が変更できるだけでなく、その列に書かれている対象スレッドＩＤに関する全てのスレッド操作権限フィールドの情報を変更することができる。

【００６３】たとえば図の例では、スレッド１（スレッドＩＤ＝１）が発したシステムコールでは、スレッド２（スレッドＩＤ＝２）の内部状態の参照と変更は可能だが、スレッド２の実行の停止／再開およびスレッド２の操作権限の変更はできないことを示している。また、スレッド１は、スレッド３（スレッドＩＤ＝３）に対しては実行の停止／再開はできるがそれ以外の操作はできない。さらに、スレッド１は、スレッド２とスレッド３以外のスレッドに対しては、何の操作もできないことを示している。また、スレッド１００（スレッドＩＤ＝１００）はスレッド２の内部状態の読み出しおよび操作権限の変更ができることを示している。

【００６４】スレッド１が対象スレッドの場合の例で、テーブル中の操作スレッドが存在するメモリ領域ＩＤの部分にａｌｌという情報が保持されているが、これは、このシステムコールがどのユーザプログラムから行われても同等で、システムコールを実行するスレッドＩＤのみに実行権限が依存することを示している。

【００６５】メモリ領域ＩＤの欄がａｌｌでないときは、そのＩＤのメモリ領域に存在するプログラムからシステムコールが実行された時の操作権限を規定する。たとえば、スレッド３が、プログラムＡ（メモリ領域ＩＤ＝Ａ）の実行中、スレッド１に対しては、すべての操作が可能であるが、それ以外のときは、操作不可である。

つまり、スレッド3がプログラムB（メモリ領域ＩＤ＝B）の実行中、スレッド1に対して同様なシステムコールを実行しても操作できないし、スレッド5（スレッドＩＤ＝5）がプログラムAにおいてスレッド1に対して同じシステムコールを実行しても操作できない。すなわち、このテーブルに記載のないスレッド操作は一切できない。

【００６６】図2の例では、スレッド操作権限変更権限のフィールドは1つだけであるが、前記3つの権限ごとに操作権の変更権限を別々に指定できるようにしても良い。

【００６７】また、スレッド操作権限には、本前の例で挙げた3つの権限の他にもあってよい。例えば、操作をするユーザのユーザＩＤ、該当フィールドを時間制限するための有効期間開始時刻、有効時間終了時刻などが考えられる。その場合にも、それら権限の変更権を設定することができる。

【００６８】また、本実施例では、スレッド操作権限とスレッド操作権の変更権限とを、同一のテーブルの中で操作スレッドＩＤ、対象スレッドＩＤ、操作スレッドが存在するメモリ領域ＩＤを共有して表現しているが、スレッド操作権限と、スレッド操作権の変更権限とを別々のテーブルに記述し、それぞれについて操作スレッドＩＤ、対象スレッドＩＤ、操作スレッドが存在するメモリ領域ＩＤを指定しても良い。

【００６９】次に、Thread_Get _Status処理部１１の処理の流れを図3に示す。

【００７０】まず、ステップＳ１にて、このシステムコールを実行した操作スレッドＩＤ、メモリ領域ＩＤをそれぞれスレッドＩＤ記憶部6とメモリ領域ＩＤ記憶部5から情報を得る。上述したように、現在実行中のスレッドＩＤは、実行状態切替部4でセットされ、現在実行中のメモリ領域ＩＤは、プロセッサの内部にあるＰＣの情報からセットされている。さらに、システムコールの第一引数から対象スレッドＩＤ検出部１３が対象スレッドＩＤを得ている。第二引数からは、スレッド情報格納位置検出部１４が、このシステムコールの実行結果を格納するユーザ・プログラムのメモリ位置を検出する。

【００７１】次にステップＳ２にて、以上の3つの情報をもとに操作権記憶部７１の表を引き、該当エントリをサーチし、スレッド操作権限を調べる。もし、見つかれば、ステップＳ３に進み、なければステップＳ８を行うエラー処理部１５へ進む。

【００７２】サーチは、テーブルの上から順に比較を行い、最初にマッチしたエントリの情報を次のステップ以降で利用する。テーブルのフィールドのａｌｌ部分は、すべてにマッチすることを意味する。もし、テーブルの最後のエントリまで比較し、マッチしなかったらサーチの失敗となる。

【００７３】ステップＳ３では、見つかったエントリの

スレッド操作権限を表す部分のうち、スレッド内部状態読出権限に該当する部分の情報を読み出し、許可か否かを判断する。この判定は操作権限判定部１２で行う。もし判定で許可されればステップＳ４へ進み、不許可になればステップＳ８のエラー処理へ移る。

【００７４】次のステップＳ４では、対象スレッドに対して、許可されたスレッドの内部状態を読み出すためのスレッド管理テーブルの参照操作を実行する。この処理はスレッド状態参照実行部１６にて行う。ここでは、該当スレッドの内部状態であるスレッド情報を、スレッド情報格納位置検出部１４で指示されたメモリ中に格納する。

【００７５】ステップＳ５にて、ステップＳ４での処理結果を判定する。処理が成功するとステップＳ６にて、成功を示すコードをシステムコール終了部１７へ伝える。失敗するとステップＳ７にて、失敗を示すコードをシステムコール終了部１７へ伝える。

【００７６】ステップＳ８では、エラーコードをシステムコール終了部１７へ伝える。

【００７７】Thread_Get _Status処理部１１での処理を終えると、最後に、システムコール終了部１７に処理が進む。このシステムコールを実行したユーザプログラムに返す返値をシステムコールの返値として戻るように処理を行い、中断していた呼出元のユーザプログラムの実行ができるように復帰処理を行い、実行レベルを特権レベルからユーザレベルへ戻す処理を行う。

【００７８】次に、操作権記憶部７１に記入されたスレッド操作権限を変更するシステムシステムコールについて説明する。

【００７９】Thread_Add _Acl(指定操作スレッドＩＤ、指定対象スレッドＩＤ、指定メモリ領域ＩＤ、追加する指定操作権限）なるシステムコールは、上記で説明したスレッド操作権限を、新たに追加するものである。

【００８０】いま、スレッド１００（スレッドＩＤ＝１００）がスレッド1（スレッドＩＤ＝1）に対し、スレッド2（スレッドＩＤ＝2）の実行の停止／再開権限を付与しようとしたとする。Ｃ言語で記載されているプログラムＣでは、次のように記されている。

【００８１】err = Thread_Add _Acl(1, 2, all, THREAD _SUSPEND);ここで、関数Thread_Add _Acl は、スレッドに対する操作権を追加するためのシステムコールであって、第1引数の1は、指定した操作スレッド番号（対象スレッドＩＤ）が1であること、すなわちスレッドＩＤ＝1に対して権限を与えることを指示している。第2引数の2は、指定した対象スレッド番号（指定対象スレッドＩＤ）が2であること、すなわちスレッドＩＤ＝2を操作する権限であることを指示している。第3引数のａｌｌは、指定した操作スレッドが存在するメモリ領域ＩＤ（指定メモリ領域ＩＤ）がａｌｌであること、すなわち操作スレッドが存在するメモリ領域ＩＤに

よって与える権限を制限することはしないということを指示している。第４引数のTHREAD_SUSPEND は、これら３つの指定した組み合わせに対して、付加する操作権限（指定操作権限）を指示している。この例では、スレッドの実行の停止／再開権限を付加することを意味している。第４引数には複数種類の権限を同時に指定することもできる。

【００８２】この関数をプログラムＣのスレッド１００が実行すると、スレッド１００はシステムコールによりプロセッサの実行モードをユーザレベルから特権レベルに遷移させ、ＯＳを呼び出す。特権レベルとは、ＯＳが処理を行う専用レベルのことである。

【００８３】呼び出されたＯＳでは、ＯＳ内のシステムコール受付部８に処理が移る。ここでは、このシステムコールを呼び出した元のユーザプログラムの中断処理を行い、要求されたシステムコールの種類と指示された引数を受け取り、それぞれその情報を引数検出部９とシステムコールの種類判別部１０に送る。

【００８４】本例では、指示されたシステムコールは、Thread_Add _Acl なので、システムコールの種類判別部１０で、Thread_Add _Acl 処理部２１を呼び出す。Thread_Add _Acl 処理部２１は図１におけるThread_Get _Status処理部１１を図４のように置き換えたものである。なお、Thread_Add _Acl 処理部２１とThread_Get _Status処理部１１は本来一体のブロック図として示すべきものであるが、図が煩雑になるため説明の便宜をはかり図１および図４の別々の図としてある。

【００８５】そこでの処理は、図２に示したスレッド操作権リスト記憶部７を使って行われる。その内容および意味は前に説明した通りである。

【００８６】次に、図５にThread_Add _Acl 処理部２１の処理の流れを示す。

【００８７】まず、ステップＳ１１にて、このシステムコールを実行した実行スレッドＩＤ、メモリ領域ＩＤをそれぞれスレッドＩＤ記憶部６とメモリ領域ＩＤ記憶部５から情報を得る。さらに、システムコールの第２引数から指定対象スレッドＩＤ検出部２３が、このシステムコールによってスレッド操作権限の変更を行う対象としたスレッドのＩＤを得る。

【００８８】次にステップＳ１２にて、以上の３つの情報をもとにスレッド操作権リスト記憶部７のテーブルを引き、該当エントリをサーチする。もし、見つかれば、ステップＳ１３に進み、なければステップＳ１９を行うエラー処理部１５へ進む。サーチは、テーブルの上から順に比較を行い、最初にマッチしたエントリの情報を次のステップ以降で利用する。テーブルのフィールドのａ１１部分は、全てにマッチすることを意味する。もし、テーブルの最後のエントリまで比較し、マッチしなかったらサーチの失敗となる。

【００８９】ステップＳ１３では、見つかったエントリのスレッド操作権変更権限に該当する部分の情報を読み出し、許可か否かを判断する。この判定は操作権変更権判定部２６で行う。もし判定で許可されればステップＳ１４へ進み、不許可になればステップＳ１９のエラー処理へ移る。

【００９０】ステップＳ１４では、システムコールの第１引数から、指定対象スレッドＩＤ検出部２３が指定した対象スレッドＩＤを得る。また、システムコールの第３引数から、指定メモリ領域ＩＤ検出部２４が指定した操作スレッドが存在するメモリ領域ＩＤを得る。さらに、システム・コールの第４引数から、指定操作権限検出部２５が指定した追加すべき操作権限を得る。

【００９１】ステップＳ１５では、ステップＳ１１およびステップＳ１４で得られた対象スレッドＩＤ、指定対象スレッドＩＤ、指定メモリ領域ＩＤ、指定操作権限をもとに操作権リスト変更部２７が操作権リスト記憶部７を書き換える。つまり、まず操作権リスト記憶部７のテーブルを順にサーチし、対象スレッドＩＤと指定対象スレッドＩＤと指定メモリ領域ＩＤの３つが全て一致する列があれば、その列のスレッド操作権限のフィールドを書き換え、もしなければ、指定した３つのＩＤを示す列を新たに操作権リスト記憶部７のテーブルに追加する。この例では、指定した３つのＩＤと一致する列が最初に見付かるので、その列のスレッド操作権限のフィールドにあるスレッドの実行の停止／再開権限を×から○に変更する。

【００９２】ステップＳ１６にて、ステップＳ１５での処理結果を判定する。処理が成功するとステップＳ１７にて、成功を示すコードをシステムコール終了部１７へ伝える。失敗するとステップＳ１８にて、失敗を示すコードをシステムコール終了部１７へ伝える。ステップＳ１９では、エラーコードをシステムコール終了部１７へ伝える。

【００９３】Thread_Add _Acl 実行部２１での処理を終えると、最後にシステムコール終了部１７に処理が進む。このシステムコールを実行したユーザプログラムに返す返値をシステムコールの返値として戻るように処理を行い、中断していた呼出元のユーザプログラムの実行ができるように復帰処理を行い、実行レベルを特権レベルからユーザレベルへ戻す処理を行う。

【００９４】本例のようなThread_Add _Acl システム・コールの実行が成功した結果、図２で示された操作権リスト記憶部７のデータは図６のように変更される。

【００９５】本例では操作権リスト変更部２７において、書換を必要とする該当フィールドを書き換えるだけで、Thread_Add _Acl システム・コールからの要求を実行することができたが、Thread_Add _Acl システム・コールに与えられた引数の内容によっては、該当フィールドを書き換えるだけでは表現できず、操作権リスト記憶部７のデータを増やさなくてはならない。

15

【００９６】例えば、操作権リスト記憶部７のデータが先の図２に示したとおりであるとき、Thread_Add _Acl (3, 2, A, THREAD_SUSPEND*THREAD_GET _STATUS) を実行した場合を考える。

【００９７】ここで、第１引数の３は、指定した操作スレッド番号（対象スレッドＩＤ）が３であること、すなわちスレッドＩＤ＝３に対して権限を与えることを指示しており、第２引数の２は、指定した対象スレッド番号（指定対象スレッドＩＤ）が２であること、すなわちスレッドＩＤ＝２を操作する権限であることを指示しており、第３引数のＡは、指定メモリ領域ＩＤがＡであること、すなわち、このシステムコールの実行により変更される権限は、プログラムＡ（メモリ領域ＩＤ＝Ａ）を実行中にのみ有効であることを指示しており、第４引数のTHREAD_SUSPEND は、これら３つの指定した組み合わせに対して、スレッドの実行の停止／再開権限を付加すること、そしてTHREAD_GET _STATUSはスレッドの内部状態読出権限を付加することを指示している。

【００９８】このシステム・コールが発せられると、操作権リスト変更部２７が、操作権リスト記憶部７のテーブルを順にサーチし、対象スレッドＩＤと指定対象スレッドＩＤと指定メモリ領域ＩＤの３つが全て一致する列をさがすが、この場合そのような列は存在しないので、このシステム・コールは、操作権リスト記憶部７の持つテーブルの一部のフィールドを書き換えるだけでは実行できない。指定した３つのＩＤを示す列を新たに操作権リスト記憶部７のテーブルに追加する。このような手順を踏むことにより、このシステム・コールの実行が終了すると、図２で示された操作権リスト記憶部７のデータは図７のように変更され、操作権リスト記憶部７の列の長さが１つ長くなっている。

【００９９】以上の説明においては、最初に操作権変更権限をどこに指定するかということについては記述されていない。一般にはＯＳの起動時に、一部の特権的なスレッドに対してこの権限を与え、以下階層的に権限を増やしてゆくことにより実現することが可能である。

【０１００】しかし、スレッドの作成時に必ず操作権変更権を指定しなければならないのは、プログラミングの手間を増やすおそれがあるので、メモリ領域またはスレッドの作成時には、あらかじめ定められたデフォルトの操作権変更権が設定されるように構成しても良い。デフォルトを何にするかについては各種考えられるが、最も簡単には、そのメモリ領域またはスレッドを起動したスレッドに対して操作権変更権を与えることが考えられる。この方法は、デフォルトとして従来ＯＳにおけるオーナーの概念を用いたことになり、さらに本実施例で説明した方法により、本発明の操作権管理装置はより柔軟な操作権管理を実現するものである。

【０１０１】また、先の例ではThread_Add _Acl の第４引数に指定することのできる指定操作権限に制約はないものとして記述したので、第４引数にTHREAD_CHANGE _ACL を指定することができる。すなわち、スレッド操作権変更権限を変更させることもできる。また、以上説明したThread_Add _Acl と同様に、Thread_DelAcl なるシステム・コールを用いて、例えば

Thread_Del _Acl(3,1,A,THREAD_CHANGE _ACL)

は、スレッドが、メモリ領域ＩＤ＝Ａのメモリ領域から、スレッドＩＤ＝１のスレッドのスレッド操作権限を取り除くことを意味している。この場合、永遠に誰にも操作権変更権限を変更することのできないようなスレッドを作ってしまうおそれがあるが、そういった変更が行われることが問題となるオペレーティングシステムにおいては、操作権変更権記憶部の情報を変更する際に、操作権記憶部の情報を変更することが不可能になるような変更を禁止するようにしても良い。例えば、先にあげた

Thread_Del _Acl(3,1,A,THREAD_CHANGE _ACL)

を実行すると、もはやスレッドＩＤ＝１のスレッドに対し、スレッド操作権限を変更することのできるスレッドが１つもなくなる。このようなシステム・コールが実行されると、操作権リスト変更部２７はエラーを検出したことになり、失敗を示すコードをシステムコール終了部１７へ伝え、操作権リスト記憶部のデータは変更されない。

【０１０２】また、操作権変更権を変更する権利を別個のものとして扱い、操作権リスト記憶部７に操作権変更権管理権限のフィールド（操作権変更権管理権限記憶部）を用意し、より確実な操作権管理を行っても良い。

【０１０３】さらには、操作権変更権記憶部の情報を変更することが不可能になるような、操作権変更権管理権限記憶部の情報の変更を禁止するようにしても良い。

【０１０４】なお、上記「スレッドＩＤ」を「ユーザＩＤ」に置き換えて構成することも可能である。この場合、図１におけるスレッドＩＤ記憶部６を、ユーザＩＤ記憶部と置き換える修正を施せば良い。

【０１０５】また、前述した「操作スレッドが存在するメモリ領域ＩＤ」を「操作スレッドが実行しているプログラムＩＤ」に置き換えて構成することも可能である。この場合、図１におけるメモリ領域ＩＤ記憶部５は、プログラムＩＤ記憶部と置き換える修正を施せば良い。

【０１０６】（第２の実施例）次に、本発明の第２の実施例について説明する。

【０１０７】本発明の操作権管理装置は、メモリ領域やスレッドに対する操作権管理を、例えばオーナーといった固定的な概念により行うのではなく、操作権変更権を操作権リスト自体に持たせ、自由に削除、追加させることによって柔軟に行うことを目的としている。ここでは、この機構を生かした例として、複数人による共同作業のアプリケーションが動作する場合に、操作権管理が柔軟に行われることを示す実施例について述べる。

【０１０８】ユーザーＡは、計算機を使った電子会議を

開催すべく、新たにスレッド１０（スレッドＩＤ＝１０）を作成した。より具体的には、ユーザーＡの動かしていたログイン・シェルであるスレッド１１がスレッド１０を作成した。このとき、操作権リスト記憶部７の該当部分のデータは図８のようになっている。以下図には操作権リスト記憶部７のテーブルのうち、本実施例を説明するのに必要な部分のみを記述し、他の部分については記述を省略する。

【０１０９】スレッド１１（スレッドＩＤ＝１１）はスレッド１０（スレッドＩＤ＝１０）の内部状態読み出しおよびスレッド操作権の変更ができる。しかし、内部状態を変更したり、実行の停止／再開を行うことはできない。つまり、既に動き始めたスレッド１０（スレッドＩＤ＝１０）は、それを作成したスレッド１１（スレッドＩＤ＝１１）によって誤って実行を妨害されることはない。これらの初期状態はスレッドを作成するシステムコールの引数として指定するものとする。

【０１１０】作成されたスレッド１０は電子会議のプログラムを実行し、ユーザーＡとのインターフェースを司る。また、スレッド１０は電子会議をサポートするためのメモリ領域として、メモリ領域３０（メモリ領域ＩＤ＝３０）を作成する。

【０１１１】メモリ領域３０に関する操作権の管理には、スレッドの操作権の管理と同様に操作権リスト記憶部７を用いる。ただし、スレッドの操作権管理の説明で述べた対象スレッドＩＤ、スレッド操作権限、スレッドの内部状態読出権限、スレッドの内部状態変更権限、スレッドの実行の停止／再開権限、スレッド操作権変更権限は、メモリ領域の操作権管理においてはそれぞれ対象メモリ領域ＩＤ、メモリ領域操作権限、メモリ領域の内部データ読出権限、メモリ領域の内部データ変更権限、メモリ領域にあるプログラムの実行権限、メモリ領域操作権変更権限、となる。

【０１１２】図９に示した操作権リスト記憶部は、スレッド１０がメモリ領域３０の内部データの読み出しおよび変更、操作権変更ができることを意味している。

【０１１３】次にユーザーＢおよびユーザーＣが電子会議への参加を表明し、スレッド１０は新たにユーザーＢおよびユーザーＣのためのスレッド１２（スレッドＩＤ＝１２）およびスレッド１３（スレッドＩＤ＝１３）を作成した。その結果、図８の操作権リスト記憶部７のデータは図１０のように追加される。また電子会議をサポートするためのメモリ領域であるメモリ領域３０は、スレッド１２およびスレッド１３からも使われるので、図９の操作権リスト記憶部７のデータは図１１のように追加される。

【０１１４】つまり、スレッド１２およびスレッド１３はスレッド１０によりコントロールされており、スレッドに関するあらゆる操作ができるが、スレッド１２およびスレッド１３は他のスレッドのコントロールをすることはできない。これにより、スレッド１０が先導して電子会議の進行をプログラムされた通りに実行することができる。

【０１１５】共通のメモリ領域として使われるメモリ領域３０は、スレッド１０、スレッド１２、スレッド１３のいずれからもデータの読み出し、変更ができるが、その領域が誰から（どのスレッドから）どのような操作ができるかということを変更することができるのはスレッド１０だけである。

【０１１６】このような仕組みで、複雑な依存関係のある電子会議のようなコミュニケーション・プログラムもＯＳの操作権管理機構を用いて確実に記述することができる。

【０１１７】次に、会議を先導していたユーザーＡのスレッド１０が、途中で会議を抜け、会議はそのまま進行している場合を考える。

【０１１８】単純にスレッド１０が自分自身の実行を終了させたのでは、メモリ領域３０の操作権を変更することのできるスレッドがいなくなり、会議を正しく進行させることができなくなる。このような場合には、ユーザーＡは一旦会議の進行権を他の人（例えばユーザーＢ）に譲らなければならない。

【０１１９】まず、メモリ領域３０の操作権をスレッド１２（ユーザーＢのスレッド、スレッドＩＤ＝１２）に設定する。そして、スレッド１３の管理権をスレッド１２に譲り、そのあとで実行を終了する。それ以外のもので、操作スレッドＩＤが１０であるものは、スレッド１０の終了と同時に自動的に消滅させる。

【０１２０】これによって、図１０および図１１の操作権リスト記憶部はそれぞれ図１２および図１３のようになる。すなわち、メモリ領域３０へのアクセスはスレッド１２とスレッド１３の両方からできるが、スレッドの管理自体の権利はスレッド１２が継承する。注意すべき点は、スレッド１０の実行が終了したために、ユーザーＡのlogin shell であるスレッド１１からは、電子会議を実行するいずれのプログラムに対しても何の権利もなくなっていることである。

【０１２１】ここに示した例は電子会議プログラムを実現する一方法を示したものであり、必ずしもこのような手順で操作権を譲渡してゆかなければプログラムが記述できないということは意味していない。

【０１２２】このようにして、スレッドおよびメモリ領域への操作権を管理しつつ、管理者の変更を実現することができる。

【０１２３】本発明の操作権管理装置を使えば、「会議に参加する」あるいは「会議から退室する」という、本実施例で具体的に例をあげて説明したような場合における、曖昧な操作権管理も、アプリケーションで逐一記述することなく、ＯＳのサポートにより柔軟かつ確実に実現することができる。

19

【０１２４】また、本発明は上述した実施例に限定されるものではなく、その要旨を逸脱しない範囲で、種々変形して実施することができる。

【０１２５】

【発明の効果】本発明の操作権管理装置によれば、スレッドまたはメモリ領域に対する操作を保護するための操作権管理を柔軟かつ確実に行うことができる。

【図面の簡単な説明】

【図１】Thread_Get _Statusを説明するための構成図

【図２】図１の操作権リスト記憶部７の内部構成例を示す図

【図３】本実施例に係わる操作権管理装置において操作権を変更する動作を示すフローチャート

【図４】Thread_Add _Acl を説明するための構成図

【図５】本実施例に係わる操作権管理装置において操作権変更権を変更する動作を示すフロー図

【図６】図４の操作権リスト記憶部７の内部構成例を示す第１の図

【図７】図４の操作権リスト記憶部７の内部構成例を示す第２の図

【図８】電子会議開催時のスレッド操作権リストを示す図

【図９】電子会議開催直後のメモリ領域操作権リストを

20

示す図

【図１０】電子会議に新たなメンバが参加した際のスレッド操作権リストを示す図

【図１１】電子会議に新たなメンバが参加した際のメモリ領域操作権リストを示す図

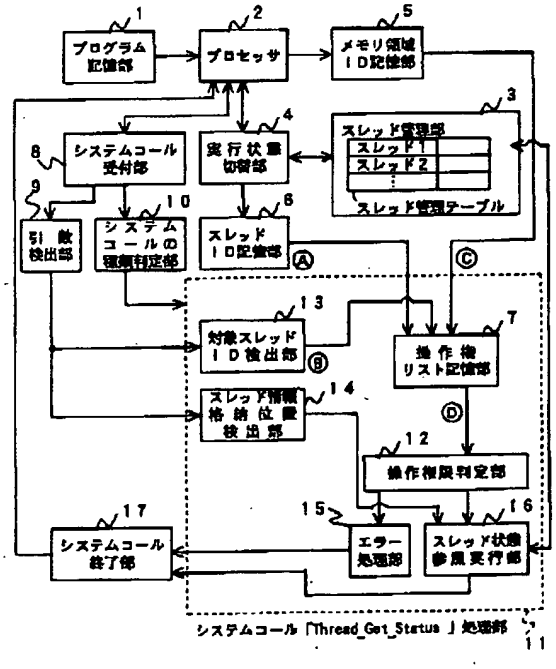
【図１２】ユーザーＡが電子会議の先導権をユーザーＢに委譲した際のスレッド操作権リストを示す図

【図１３】ユーザーＡが電子会議の先導権をユーザーＢに委譲した際のメモリ領域操作権リストを示す図

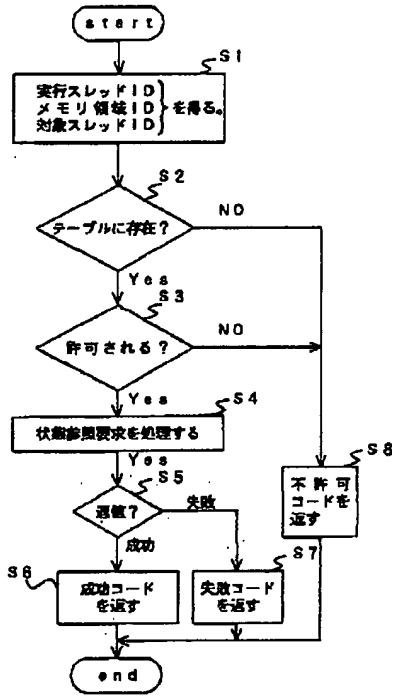
【符号の説明】

１…プログラム記憶部、２…プロセッサ、３…スレッド管理部、４…実行状態切替部、５…メモリ領域ＩＤ記憶部、６…スレッドＩＤ記憶部、７…操作権リスト記憶部、８…システムコール受付部、９…引数検出部、１０…システムコールの種類判定部、１１，２１…システムコール実行部、１２…操作権限判定部、１３…対象スレッドＩＤ検出部、１４…スレッド情報格納位置検出部、１５…エラー処理部、１６…スレッド状態参照実行部、１７…システムコール終了部、２２…指定操作スレッドＩＤ検出部、２３…指定対象スレッドＩＤ検出部、２４…指定メモリ領域ＩＤ検出部、２５…指定操作権限検出部、２６…操作権変更権判定部、２７…操作権リスト変更部

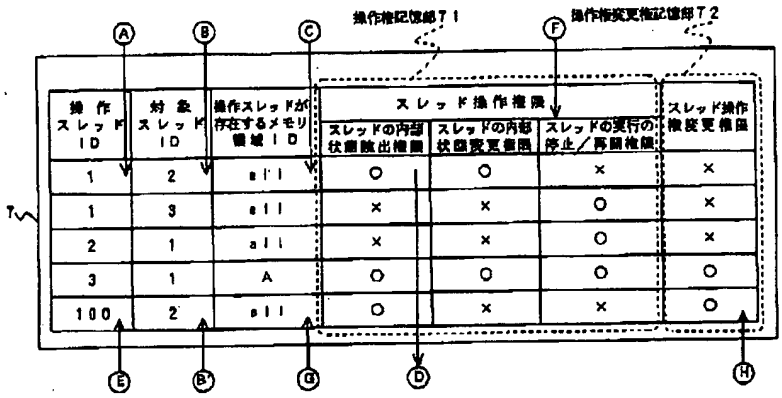
【図１】



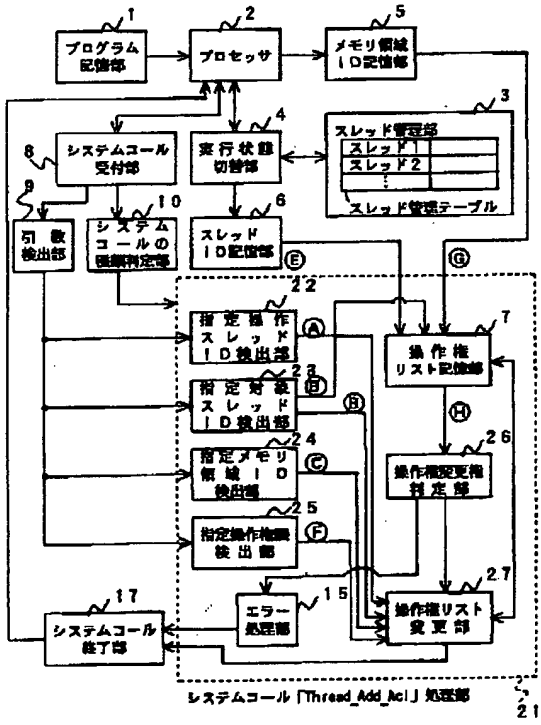
【図３】



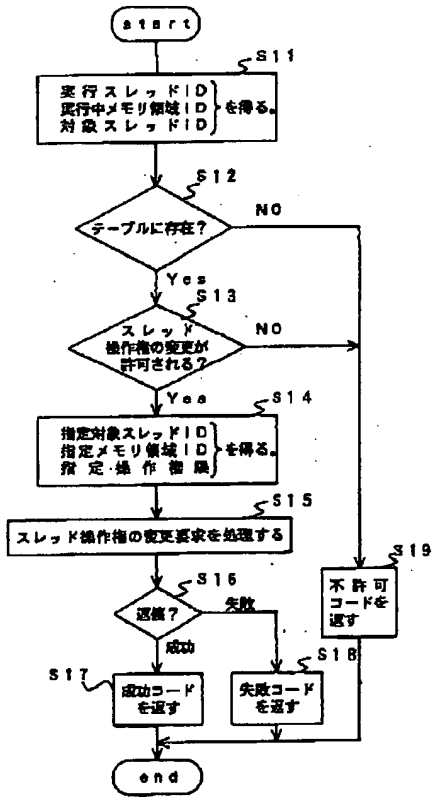
【図2】



【図4】



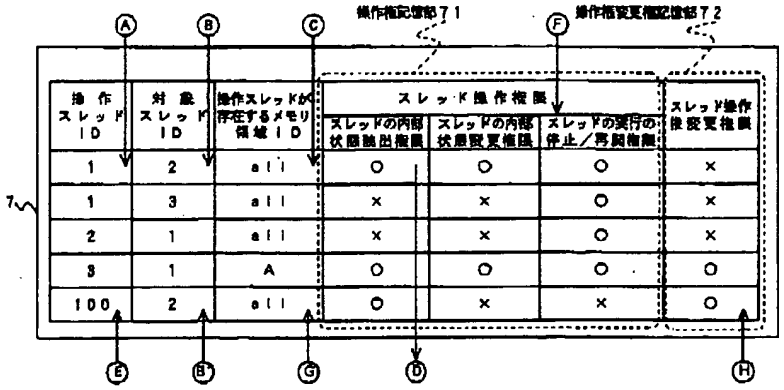
【図5】



【図9】

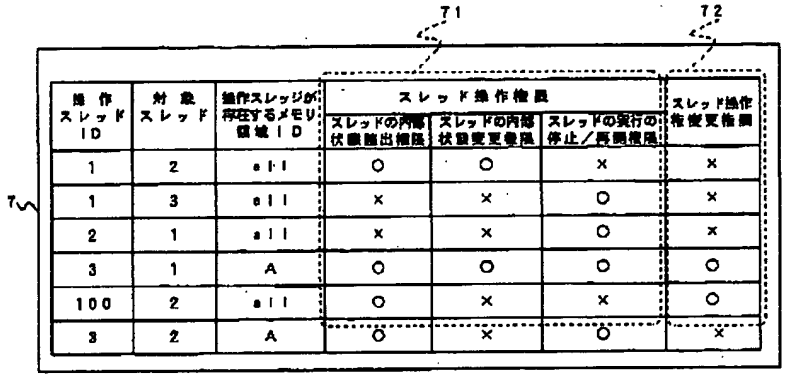
操作スレッドID	対象メモリ領域ID	操作スレッドが存在するメモリ領域ID	メモリ領域操作権限			メモリ領域操作権変更権限
			メモリ領域の内部データ読出権限	メモリ領域の内部データ変換権限	メモリ領域のプログラム実行権限	
10	30	all	○	○	×	○

【図6】



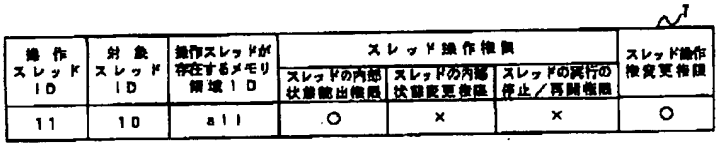
操作スレッドID	対象スレッドID	操作スレッドが存在するメモリ領域ID	スレッド操作権限			スレッド操作権変更権限
			スレッドの内部状態読出権限	スレッドの内部状態変更権限	スレッドの実行の停止／再開権限	
1	2	all	○	○	○	×
1	3	all	×	×	○	×
2	1	all	×	×	○	×
3	1	A	○	○	○	○
100	2	all	○	×	×	○

【図7】



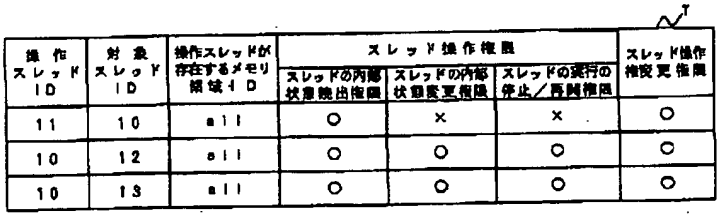
操作スレッドID	対象スレッド	操作スレッジが存在するメモリ領域ID	スレッド操作権限			スレッド操作権変更権限
			スレッドの内部状態読出権限	スレッドの内部状態変更権限	スレッドの実行の停止／再開権限	
1	2	all	○	○	×	×
1	3	all	×	×	○	×
2	1	all	×	×	○	×
3	1	A	○	○	○	○
100	2	all	○	×	×	○
3	2	A	○	×	○	×

【図8】



操作スレッドID	対象スレッドID	操作スレッドが存在するメモリ領域ID	スレッド操作権限			スレッド操作権変更権限
			スレッドの内部状態読出権限	スレッドの内部状態変更権限	スレッドの実行の停止／再開権限	
11	10	all	○	×	×	○

【図10】



操作スレッドID	対象スレッドID	操作スレッドが存在するメモリ領域ID	スレッド操作権限			スレッド操作権変更権限
			スレッドの内部状態読出権限	スレッドの内部状態変更権限	スレッドの実行の停止／再開権限	
11	10	all	○	×	×	○
10	12	all	○	○	○	○
10	13	all	○	○	○	○

【図１１】

操作スレッドID	対象メモリ領域ID	操作スレッドが存在するメモリ領域ID	メモリ領域操作権限			メモリ領域操作権変更権限
			メモリ領域の内部データ読出権限	メモリ領域の内部データ変更権限	メモリ領域のプログラム実行権限	
10	30	all	○	○	×	○
12	30	all	○	○	×	×
13	30	all	○	○	×	×

【図１２】

操作スレッドID	対象スレッドID	操作スレッドが存在するメモリ領域ID	スレッド操作権限			スレッド操作権変更権限
			スレッドの内部状態読出権限	スレッドの内部状態変更権限	スレッドの実行の停止／再開権限	
12	13	all	○	○	○	○

【図１３】

操作スレッドID	対象メモリ領域ID	操作スレッドが存在するメモリ領域ID	メモリ領域操作権限			メモリ領域操作権変更権限
			メモリ領域の内部データ読出権限	メモリ領域の内部データ変更権限	メモリ領域のプログラム実行権限	
12	30	all	○	○	×	○
13	30	all	○	○	×	×

フロントページの続き

(72)発明者　申　承昊
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

(72)発明者　吉田　英樹
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内